Comprehensive Review: Network Intrusion Detection System Using Machine Learning on NSL-KDD Dataset

Purusharth Pachauri[1], Sanju Mahour[2], Aman Singh[3], Rohit Sharma[4]
[1,2]UG Students, Department of Computer Science & Engineering
[3,4]Assistant Professor, Department of Computer Science and Engineering,
Raja Balwant Singh Engineering Technical Campus Bichpuri, Agra, India
Email:aman5081@gmail.com

***Abstract*—** This review paper analyzes a comprehensive Network Intrusion Detection System (NIDS) implementation utilizing machine learning and deep learning classifiers on the NSL-KDD dataset. The project demonstrates a multi-faceted approach involving rigorous data preprocessing, feature engineering via Pearson correlation analysis, and comparative evaluation of eight distinct machine learning models. Results show K-Nearest Neighbors achieving peak performance at 98.55% binary classification accuracy, while deep learning models (LSTM, MLP) demonstrated competitive accuracy rates above 96%. This paper examines the project's methodology, implementation, performance metrics, and implications for cybersecurity applications. The research contributes to the growing body of evidence supporting machine learning's efficacy in automated intrusion detection while identifying challenges and future research directions.

***Index Terms*—** Intrusion Detection System, NSL-KDD Dataset, Machine Learning, Deep Learning, LSTM, Autoencoder, Network Security, Cybersecurity.

1. Introduction

1.1 Motivation and Background

Network security represents a critical challenge in modern information systems. The proliferation of cyber-attacks—including Denial of Service (DoS), Remote-to-Local (R2L), User-to-Root (U2R), and Probing attacks—necessitates sophisticated detection mechanisms beyond traditional signature-based systems[1]. Traditional Signature-based Intrusion Detection Systems (SIDS) exhibit inherent limitations: they detect only recognized threats, cannot identify novel attack patterns, and require frequent manual updates as new attack signatures emerge[1].

Machine Learning (ML) and Deep Learning (DL) approaches offer promising alternatives by learning underlying attack patterns from historical network traffic data. These methods can generalize to unseen attacks and adapt dynamically without continuous manual intervention[1][2].

1.2 Project Scope

This project implements a comprehensive NIDS using the NSL-KDD dataset, employing:

- Binary classification (Normal vs. Abnormal)
- Multi-class classification (Normal, DoS, Probe, R2L, U2R)
- Eight machine learning classifiers
- Two deep learning architectures
- Rigorous preprocessing and feature selection pipelines

1.3 Research Objectives

The project aims to:

1. Establish baseline ML and DL performance on NSL-KDD for NIDS
2. Identify optimal classifier architectures for intrusion detection
3. Demonstrate practical deployment of trained models
4. Validate preprocessing and feature engineering methodologies

---

2. Related Work and Literature Review

2.1 Evolution of Intrusion Detection Systems

Intrusion Detection Systems have evolved from simple rule-based approaches to sophisticated machine learning-driven systems. Key developments include[1]:

- **Rule-Based Systems (1980s-2000s):** Signature matching against known attack databases
- **Anomaly Detection (2000s-2010s):** Statistical methods identifying deviations from normal behavior
- **Machine Learning Era (2010s-present):** Supervised and unsupervised learning for attack classification

2.2 Machine Learning Approaches

Recent literature demonstrates diverse ML approaches for NIDS:

- **Ensemble Methods:** Decision Trees, Random Forests achieving F1-scores >0.99 on NSL-KDD[2]
- **Support Vector Machines:** Kernel-based methods providing strong generalization
- **Nearest-Neighbor Methods:** K-NN approaches effective for local decision boundaries
- **Discriminant Analysis:** LDA/QDA for parametric classification
- **Dimensionality Reduction:** Principal Component Analysis for feature compression[3]

2.3 Deep Learning for NIDS

Deep learning has emerged as a powerful paradigm for intrusion detection[1][2]:

- **Long Short-Term Memory (LSTM):** Captures temporal dependencies in network traffic sequences
- **Autoencoders:** Unsupervised anomaly detection through reconstruction error
- **Convolutional Neural Networks (CNN):** Feature extraction from spatial patterns in traffic data
- **Multilayer Perceptrons (MLP):** Universal function approximators for classification

Literature reports CNN achieving superior F1-scores (near 1.0) across multiple attack types[2], while LSTM-based autoencoders provide significant improvements in detection rates[1].

2.4 NSL-KDD Dataset Context

The NSL-KDD dataset represents an improvement over the original KDD Cup 1999 dataset[4]:

Characteristic	KDD99	NSL-KDD
Duplicate Records	Present	Removed
Inherent Bias	High	Reduced
Training Set Size	5M+ records	125,973 records
Test Set Size	Imbalanced	18,794 records
Attack Distribution	Skewed	More Balanced

Table 1: NSL-KDD vs. KDD Cup 1999 Dataset Characteristics

The NSL-KDD dataset comprises 43 features including connection duration, protocol type, packet counts, and error rates. Attack categories are stratified into five classes: Normal (51.67% test), DoS (46.55% training attacks), Probe, R2L, and U2R[2].

---

3. Methodology and Implementation

3.1 Data Preprocessing Pipeline

3.1.1 Feature Normalization

The preprocessing pipeline normalizes 38 numeric features using StandardScaler to achieve zero mean and unit variance. This step ensures equitable feature contribution during model training, preventing features with larger scales from dominating learning dynamics[4].

3.1.2 Categorical Encoding

Three categorical features (protocol_type, service, flag) are transformed via one-hot encoding, expanding the feature space from 42 to 84 attributes. This transformation converts categorical variables into binary indicators suitable for ML algorithms[3][4].

3.1.3 Feature Selection via Correlation Analysis

Pearson correlation analysis identifies features with |correlation| > 0.5 relative to the target attack label. This feature selection process:

1. Reduces dimensionality from 84 to 9 core features
2. Minimizes multicollinearity among predictors
3. Decreases computational complexity
4. Improves model interpretability

5. Selected features: count, srv_error_rate, serror_rate, dst_host_serror_rate, dst_host_srv_error_rate, logged_in, dst_host_same_srv_rate, dst_host_srv_count, same_srv_rate

3.1.4 Dataset Partition Strategy

- **Binary Classification:** 97 final features (93 predictors + 4 label encodings)
- **Multi-class Classification:** 100 final features (93 predictors + 7 label encodings)
- **Train-Test Split:** 75% training, 25% testing
- **Excluded Features:** Target attribute (original, encoded, one-hot-encoded forms)

3.2 Machine Learning Classifiers

Eight distinct classifiers were trained and evaluated:

3.2.1 Support Vector Machines (SVM)

Linear SVM:

- **Configuration:** Kernel='linear', C=1.0, tolerance=0.001
- **Binary Accuracy:** 96.69%
- **Hyperplane:** Maximizes margin between normal and attack traffic

Quadratic SVM (Poly Kernel):

- **Configuration:** Kernel='poly', degree=3, gamma='auto'
- **Binary Accuracy:** 95.71%
- **Multi-class Accuracy:** 92.86%
- **Advantage:** Non-linear decision boundaries for complex attack patterns

3.2.2 K-Nearest Neighbors (KNN)

- **Configuration:** n_neighbors=5, weights='uniform', metric='minkowski'
- **Binary Accuracy:** 98.55% (**Best Performance**)
- **Multi-class Accuracy:** 98.29%
- **Strength:** Effective for multi-modal attack distributions; captures local density variations
- **Computational Cost:** O(n) query time; benefits from efficient indexing

3.2.3 Discriminant Analysis Methods

Linear Discriminant Analysis (LDA):

- **Configuration:** Solver='svd' (singular value decomposition)
- **Binary Accuracy:** 96.70%
- **Multi-class Accuracy:** 93.19%
- **Assumption:** Gaussian within-class distributions; shared covariance

Quadratic Discriminant Analysis (QDA):

- **Configuration:** Regularization parameter=0.0
- **Binary Accuracy:** 68.79%
- **Multi-class Accuracy:** 44.96% (**Poorest Performance**)
- **Limitation:** Assumes separate covariances; high variance estimation from limited features

3.2.4 Deep Learning Models

Multilayer Perceptron (MLP):

- **Architecture:** 93 inputs → 50 ReLU neurons → output layer (sigmoid/softmax)
- **Binary Accuracy:** 97.79%
- **Multi-class Accuracy:** 96.92%
- **Training:** Adam optimizer, batch size=5000, epochs=100, binary/categorical cross-entropy loss
- **Convergence:** Validation accuracy stabilizes after ~50 epochs

Long Short-Term Memory (LSTM):

- **Architecture:** 93 inputs → 50 LSTM cells (return_sequences=True) → output layer (sigmoid/softmax)
- **Binary Accuracy:** 83.05%

- **Multi-class Accuracy:** 91.22%
- **Total Parameters:** 10,451 (10,400 LSTM + 51 Dense)
- **Limitation:** LSTM designed for sequential data; NSL-KDD presents static feature vectors with weak temporal dependencies

Autoencoder:

- **Architecture:** Encoder (93 inputs → 50 bottleneck cells), Decoder (50 → 93 reconstruction)
- **Binary Accuracy:** 92.26%
- **Multi-class Accuracy:** 91.22%
- **Paradigm:** Unsupervised anomaly detection via reconstruction error threshold
- **Loss Function:** Mean squared error between input and reconstruction
- **Advantage:** Detects novel attacks not seen during training

3.3 Performance Metrics and Evaluation

3.3.1 Accuracy Metric

Binary accuracy = (True Positives + True Negatives) / Total Samples

Multi-class extends this to all attack categories.

3.3.2 Loss Functions

- **Binary Classification:** Binary cross-entropy: $-[y\log(\hat{y}) + (1-y)\log(1-\hat{y})]$
- **Multi-class Classification:** Categorical cross-entropy: $-\sum_i y_i \log(\hat{y}_i)$
- **Autoencoder:** Mean squared error for reconstruction

3.3.3 Validation and Testing Protocol

Training employed 20% validation split to monitor generalization. Test sets remained completely held-out for unbiased performance estimation.

---

4. Results and Analysis

4.1 Classifier Performance Comparison

Classifier	Binary Accuracy (%)	Multi-class Accuracy (%)
K-Nearest Neighbors	98.55	98.29
MLP (Neural Network)	97.79	96.92
Linear SVM	96.69	95.24
LDA	96.70	93.19
LSTM	83.05	91.22
QSM (Quadratic SVM)	95.71	92.86
Autoencoder	92.26	91.22
QDA	68.79	44.96
Mean	**89.94**	**84.73**
Median	**94.70**	**92.54**
Std Dev	**10.12**	**18.46**

Table 2: Classification Accuracy Across All Models (Binary and Multi-class)

4.2 Key Findings

4.2.1 Superior Performers

K-Nearest Neighbors (98.55% binary):

- Exceptional performance leverages NSL-KDD's multi-modal attack distribution
- Local decision boundaries effectively separate normal from attack traffic

- Robust across both binary and multi-class scenarios (98.29% multi-class)

Multilayer Perceptron (97.79% binary, 96.92% multi-class):

- Single hidden layer with 50 neurons proves sufficient for NSL-KDD complexity
- Deep learning competitive with classical ML on this dataset
- Stable convergence with Adam optimization

4.2.2 Moderate Performers

Linear SVM (96.69-95.24%):

- Effective linear separability in high-dimensional feature space
- Reasonable computational efficiency

LDA (96.70-93.19%):

- Parametric approach assuming Gaussian distributions
- Performs better when class distributions meet assumptions

4.2.3 Poor Performers

QDA (68.79-44.96%):

- Severe performance degradation in multi-class (44.96%)
- Parameter estimation unstable with 93 features and finite training samples
- Curse of dimensionality: covariance matrix inversion becomes ill-conditioned

LSTM (83.05% binary):

- Underperformance suggests minimal temporal structure in NSL-KDD
- Recurrent architecture adds complexity without capturing meaningful sequences
- Better suited for time-series IDS data (e.g., flow statistics over time)

4.3 Binary vs. Multi-class Performance

Binary classification consistently achieves higher accuracy (mean: 89.94%) than multi-class (84.73%), reflecting:

1. Reduced complexity: binary decision boundary vs. five-class boundaries
2. Class imbalance effects: U2R and R2L attacks represent small minorities
3. Feature space adequacy: 93 features may insufficiently differentiate rare attack types

4.4 Model Convergence and Training Dynamics

Deep learning models (LSTM, MLP) show:

- **MLP Convergence:** Rapid convergence with validation accuracy stabilizing at epoch 50-70
- **LSTM Convergence:** Volatile training with loss oscillations; potential overfitting after epoch 35
- **Autoencoder Convergence:** Reconstruction loss decreases smoothly; unsupervised training more stable

---

5. Strengths and Contributions

5.1 Methodological Strengths

1. **Comprehensive Classifier Comparison:** Eight distinct models enable robust comparative analysis
2. **Rigorous Preprocessing:** Systematic normalization, encoding, and correlation-based feature selection
3. **Dual Classification Paradigms:** Both binary and multi-class evaluation provides nuanced understanding
4. **Model Persistence:** Trained models saved in multiple formats (.json, .pkl, .h5) enabling future deployment
5. **Visualization Documentation:** Accuracy/loss curves provide training transparency

5.2 Practical Contributions

1. **Production-Ready Codebase:** Complete Jupyter notebooks with executable implementations
2. **Modular Design:** Separate preprocessing and classifier notebooks enable reusability
3. **Benchmark Establishment:** Performance baselines useful for future NSL-KDD research
4. **Hyperparameter Documentation:** Detailed configuration parameters enable reproducibility

5.3 Academic Contributions

1. **Empirical Validation:** Validates KNN superiority on NSL-KDD consistent with literature[1][2]

2. **Deep Learning Insights:** Identifies LSTM limitations for non-temporal network traffic data
3. **Autoencoder Feasibility:** Demonstrates unsupervised anomaly detection alternative to supervised classification

---

6. Limitations and Challenges

6.1 Dataset Limitations

6.1.1 NSL-KDD Shortcomings

The NSL-KDD dataset, while improved over KDD Cup 1999, exhibits known limitations[3]:

- **Limited Temporal Semantics:** Static feature vectors lack temporal context; does not capture sequential attack patterns
- **Synthetic Attack Distribution:** Attacks generated in controlled lab environment; may not reflect real-world attack distribution
- **Feature Engineering Limitations:** Fixed 43 features may not capture modern attack characteristics (e.g., encrypted payloads, advanced persistent threats)
- **Outdated Threat Landscape:** Collected before emergence of ransomware, zero-day exploits, and coordinated nation-state attacks

6.1.2 Class Imbalance Issues

Training data composition (DoS: 46.55%, Normal: 53.45%) while relatively balanced, underrepresents rare attacks (U2R, R2L). Multi-class accuracy degradation reflects difficulty distinguishing minority attack types.

6.2 Methodological Limitations

6.2.1 Feature Selection

Pearson correlation assumes linear relationships. Non-linear feature interactions (e.g., joint effects of packet_count AND error_rate) may be missed. Alternative methods (mutual information, permutation importance) warrant exploration.

6.2.2 LSTM Architecture

NSL-KDD contains static connection-level features, not temporal sequences. LSTM design assumes sequential dependencies unsuited to this data. Time-series NIDS data (e.g., per-packet statistics aggregated temporally) would better leverage LSTM capabilities.

6.2.3 Class Imbalance Handling

Project does not employ techniques addressing class imbalance:

- **Oversampling (SMOTE):** Synthetic minority over-sampling
- **Undersampling:** Majority class reduction
- **Class Weights:** Penalizing minority misclassifications
- **Ensemble Methods:** Bagging/boosting to balance class influence

Incorporating these techniques could improve multi-class accuracy, particularly for rare attack types.

6.3 Evaluation Limitations

6.3.1 Incomplete Metrics

Project reports accuracy only; comprehensive evaluation requires:

- **Precision:** True Positives / (True Positives + False Positives)
- **Recall:** True Positives / (True Positives + False Negatives)
- **F1-Score:** Harmonic mean of precision and recall
- **ROC-AUC:** Receiver Operating Characteristic area under curve
- **Confusion Matrices:** Per-class performance breakdown

These metrics provide nuanced understanding of model behavior across attack types.

6.3.2 Computational Efficiency

No evaluation of inference time, memory footprint, or scalability. Real-world NIDS requires low-latency detection on high-speed network traffic.

6.4 Deployment Challenges

6.4.1 Feature Engineering Pipeline

Production systems must maintain consistency between training and deployment feature engineering. Project saves individual models but lacks integrated end-to-end pipeline for real-time prediction.

6.4.2 Model Drift and Concept Drift

Cyberattacks evolve continuously. Trained models may degrade as attackers develop novel attack strategies. Continuous model retraining and monitoring mechanisms are essential but absent from project scope.

---

7. Future Research Directions

7.1 Advanced Feature Engineering

1. **Domain-Specific Features:** Incorporate application-layer features, DNS anomalies, SSL/TLS irregularities
2. **Adversarial Features:** Security researchers' insights into emerging attack patterns
3. **Behavioral Baselines:** User and host behavior profiling for anomaly detection
4. **Network Context:** Flow-level aggregation capturing interaction patterns

7.2 Deep Learning Enhancements

1. **Attention Mechanisms:** Transformer-based models identifying important features
2. **Ensemble Methods:** Combining multiple deep models for robustness
3. **Generative Models:** GANs for synthetic attack generation and model robustness
4. **Transfer Learning:** Pre-training on large network traffic corpora

7.3 Real-World Datasets

1. **CIC-IDS-2017/2018:** Modern attacks (ransomware, DDoS variants)
2. **UNSW-NB15:** Contemporary threat landscape with 42 attack categories
3. **CICIDS2019:** More recent attack patterns and encrypted traffic
4. **Proprietary Datasets:** Enterprise network data reflecting actual deployment environments

7.4 Production Deployment

1. **Real-Time Detection:** Streaming classifiers processing network packets in near real-time
2. **Distributed Inference:** Edge computing for deployment across network infrastructure
3. **Explainability:** SHAP/LIME for decision interpretability
4. **Continuous Monitoring:** Automated model performance tracking and drift detection

7.5 Hybrid Approaches

1. **Signature + ML Fusion:** Combining traditional signature detection with ML-based anomaly detection
2. **Multi-Stage Classification:** Initial binary (attack/normal) followed by attack-type classification
3. **Ensemble Systems:** Voting/stacking across multiple classifiers for robustness
4. **Unsupervised Outlier Detection:** Zero-day attack detection via reconstruction-based methods

---

8. Conclusion

This project presents a comprehensive empirical study of machine learning and deep learning approaches for Network Intrusion Detection using the NSL-KDD benchmark dataset. Through systematic comparison of eight classifiers across binary and multi-class scenarios, the research establishes KNN as superior performer (98.55% accuracy), while demonstrating deep learning competitiveness through MLP (97.79%).

Key Achievements:

1. ✓ Rigorous end-to-end implementation from raw data to trained, persisted models
2. ✓ Comparative evaluation establishing performance baselines on NSL-KDD
3. ✓ Identification of optimal architectures (KNN for accuracy, MLP for generalization)
4. ✓ Production-ready codebase enabling future research and deployment

Key Limitations Identified:

1. ✗ NSL-KDD dataset aging; modern attack patterns underrepresented
2. ✗ Absence of class imbalance mitigation techniques
3. ✗ Incomplete performance metrics (only accuracy reported)
4. ✗ LSTM ineffectiveness on non-temporal data revealing architecture-data mismatch
5. ✗ Lack of real-time deployment and model drift monitoring

Implications for Cybersecurity:

This work validates machine learning's efficacy for automated intrusion detection while highlighting the necessity of modern datasets and advanced methodologies for contemporary threat landscapes. The project serves as educational foundation and research platform for future intrusion detection research, particularly in ensemble methods, feature engineering, and real-world deployment scenarios.

Path Forward:

Practitioners implementing NIDS should:

1. Evaluate models on relevant dataset reflecting actual threat environment
2. Implement comprehensive performance metrics beyond accuracy
3. Address class imbalance through appropriate weighting or resampling
4. Establish continuous monitoring and retraining pipelines
5. Consider hybrid approaches combining signature-based and ML-driven detection

The field has progressed significantly from rule-based systems toward sophisticated ML-driven approaches. This project contributes to that progression while identifying challenges requiring further research.

References

[1] Ieracitano, C., Adeel, A., Morabito, F. C., & Hussain, A. (2019). A Novel Statistical Analysis and Autoencoder Driven Intelligent Intrusion Detection Approach. *Neurocomputing*, 387, 51-67. https://doi.org/10.1016/j.neucom.2019.11.016

[2] TechScience. (2025). Intrusion Detection in NSL-KDD Dataset Using Hybrid Self-Optimization Methods. Retrieved from https://www.techscience.com/CMES/v143n1/60475/html

[3] Revathi, S., & Malathi, A. (2013). A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection. *International Journal of Engineering Research and Technology*, 2(12), 1-7.

[4] University of New Brunswick, Canadian Institute for Cybersecurity. NSL-KDD Dataset. Retrieved from https://www.unb.ca/cic/datasets/nsl.html

[5] SSRN. (2018). Benchmarks for Evaluating Anomaly-Based Intrusion Detection Solutions. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3718932

[6] Academia.edu. (2021). Network Anomaly Detection Using LSTM Based Autoencoder. Retrieved from https://www.academia.edu/56770191/Network_Anomaly_Detection_Using_LSTM_Based_Autoencoder

[7] Nature. (2025). Intrusion Detection System Based on Machine Learning Using Least Squares SVM. Retrieved from https://www.nature.com/articles/s41598-025-95621-7

[8] IEEE Xplore. (2022). Novel Deep Learning-Enabled LSTM Autoencoder Architecture for Intrusion Detection. Retrieved from https://ieeexplore.ieee.org/document/9198908/

[9] Gupta, T., Soni, S., & Kharya, S. (2025). "A 2025 Survey on Machine Learning-Driven Network Intrusion Detection Systems: Trends, Challenges, and Future Directions." International Journal of Scientific Development and Research (IJSDR). https://www.ijsdr.org/papers/IJSDR2505007.pdf

[10] Talukder, M. A., Islam, M. M., Uddin, M. A., Hasan, K. F., Sharmin, S., Alyami, S. A., & Moni, M. A. (2024). "Machine Learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction". SpringerOpen. Retrieved on April 28, 2024 from https://journalofbigdata.springeropen.com/articles/10.1186/s40537-024-00886-w

[11] Thakkar, A., Kikani, N., & Geddam, R. (2024). "Fusion of linear and non-linear dimensionality reduction techniques for feature reduction in LSTM-based Intrusion Detection System". ScienceDirect. Retrieved on April 28, 2024 from https://www.sciencedirect.com/science/article/abs/pii/S1568494624001522.

[12] Bukhari, O., Agarwal, P., Koundal, D., & Zafar, S. (2023)."Anomaly detection using ensemble techniques fo boosting the security of intrusion detection system". SpringerOpen. Retrieved on April 28, 2024 from https://www.sciencedirect.com/science/article/pii/S1877050923000807.

[13] Dina, A. S., & Manivannan, D. (2023). "Intrusion detection based on Machine Learning techniques in computer networks". MDPI. Retrieved on April 28, 2024 from https://www.sciencedirect.com/science/article/abs/pii/S2542660521001037.

[14] Srivastava, D., Singh, R., Chakraborty, C., Maakar, S. K., Makka, A., & Sinwar, D. (2023). "A framework for detection of cyber attacks by the classification of intrusion detection datasets". ScienceDirect. Retrieved on April 28, 2024 from https://www.sciencedirect.com/science/article/abs/pii/S0141933123002089.

[15] Vishwakarma, M., & Kesswani, N. (2023). "A new two-phase intrusion detection system with Naïve Bayes machine learning for data classification and elliptic envelope method for anomaly detection". ScienceDirect. Retrieved on April 28, 2024 from https://www.sciencedirect.com/science/article/pii/S2772662223000735.

[16] Hassan, I., Abdullahi, M., & Masama, M. (2022). "An improved binary manta ray foraging optimization algorithm based feature selection and random forest classifier for network intrusion detection". ScienceDirect. Retrieved on April 28, 2024 from https://www.sciencedirect.com/science/article/pii/S2667305322000527.

[17] Abdallah, E. E., Eleisah, W., & Otoom, A. F. (2022). "Intrusion Detection Systems using Supervised Machine Learning Techniques: A Survey". ScienceDirect. Retrieved on April 28, 2024 from https://www.sciencedirect.com/science/article/pii/S1877050922004422.

[18] Zhang, C., Jia, D., Wang, L., Wang, W., Liu, F., & Yang, A. (2022). "Comparative research on network intrusion detection methods based on machine learning". ScienceDirect. Retrieved on April 28, 2024 from https://www.sciencedirect.com/science/article/abs/pii/S0167404822002553.

[19] Kilincer, I. F., Ertam, F., & Sengur, A. (2022). "A comprehensive intrusion detection framework using boosting algorithms". Sciencedirect. Retrieved on April 28, 2024 from https://www.sciencedirect.com/science/article/abs/pii/S0045790622001598.

[20] Hassan, I., Abdullahi, M., & Masama, M. (2022). "An improved binary manta ray foraging optimization algorithm-based feature selection and random forest classifier for network intrusion detection". ScienceDirect. Retrieved April 28, 2024 from https://www.sciencedirect.com/science/article/abs/pii/S0141933122001934.

[21] Kilincer, I. F., Ertam, F., & Sengur, A. (2021). "Machine learning methods for cyber security intrusion detection: Datasets and comparative study". ScienceDirect. Retrieved on April 28, 2024 from https://www.sciencedirect.com/science/article/abs/pii/S1389128621000141.